OMNIMON! (TM)
USER'S GUIDE

by

David Young, CDY Consulting

SYSTEM REQUIREMENTS: ATARI 800/400 Home Computer

AUTHOR'S EARNEST ENTREATY

I have done my best to offer here a quality program at a reasonable price. This is my livelihood. Please do not make copies of this program for any reason other than personal backup. Thank you.

INTRODUCTION

Greetings fellow ATARI Home Computer owner. I am sure you are just as proud of your system as I am of mine and enjoy buying accessories to extend its power and convenience. From that point of view, OMNIMON! is one of the most powerful additions you can make to your computer. Any serious ATARI owner will find it indispensable after using it for the first time.

OMNIMON! is a resident machine language monitor which, once installed, is always available to you. What that means is that you never have to load it and you can call it up no matter what program happens to be running at the time. Once running, OMNIMON! gives you complete control over your computer. This includes the ability to easily examine and modify memory or the 6502's registers, to dump data to a printer, and to read and write to the disk drive(s) without DOS. It also has a complete set of debugging tools including a disassembler, single step, and a unique JSR function for testing out subroutines. And all of these features are available to you at any time, no matter what program is running, simply by pressing SYSTEM RESET along with either the OPTION or SELECT button!

If you have been wanting to learn assembly language programming, OMNIMON! can make it a very pleasant experience. Since it is ROM resident, you can always get back to OMNIMON! even if your program hangs up in an infinite loop. (See the description of OMNIMONA, the advanced version of the monitor, for a method of recovery if the system is locked up.) You can even call OMNIMON! at critical points in your program to examine things before continuing execution. OMNIMON! is extremely user friendly so even programmers with little experience should find it very easy to use.

GETTING STARTED

After installing OMNIMON! in your computer (if you have not done so, see OMNIMON! INSTALLATION INSTRUCTIONS), you should be able to powerup your computer as usual. To enter the OMNIMON! program, hold down the OPTION key and press SYSTEM RESET. This method of entering OMNIMON! will cause a warmstart up to the point that the application program would normally be given control. Instead OMNIMON! takes control and you should see the OMNIMON! header written across the top of the screen indicating that the program is running:

```
        David Young OMNIMON! Copyright 1983

  PC NV-BDIZC ACCUM X-REG Y-REG STACK
Axxxx   xx     xx    xx    xx    xxx
```

When you are ready to exit OMNIMON!, hold down the START button and type RETURN. This will cause the warmstart to go to completion, giving control back to the application program. Notice that pressing SYSTEM RESET by itself will cause a normal warmstart. Later we will discuss another method of entering OMNIMON! which preserves the PC and CPU registers of the interrupted program.

Once you have OMNIMON! running, the first command to learn is the 'HELP' command. As is fairly standard practice in user friendly software, OMNIMON! uses '?' (RETURN) to give you a list of all the commands available. So type '?' followed by RETURN and you will see the following:

```
CPU/CHG:C
DPY/CHG:D (adr adr)
EXECUTE:E (byt)
JSR    :J adr
LINK DR:L (drive#)
PRINTER:P
RD DISK:R (sec# adr #)
SEARCH :S adr byt byt ...
TOGGLE :T
WR DISK:W (sec# adr #)
DIS/CHG:X (adr adr)
PSH STK:+ byt byt ...
POP STK:-
```

The HELP command not only provides a list of commands but also indicates the parameters each command expects. Parameters in parentheses are optional. If they are omitted, OMNIMON! will try to interpret the command in a manner convenient to you. Usually this means executing the command on the next logical memory location or sector. If you are anxious to start using OMNIMON!, you can do so immediately with just this little bit of knowledge. With a little experimentation you should have little trouble figuring out what most of the commands do. When you are ready to learn some of the more subtle features built into OMNIMON!, read the rest of this documentation.

There are a few important things to point out before proceeding:

1) All numerical input and output is done in hex.

2) Parameters are delimited by a space or other non-hex character.

3) The command being processed will be aborted if an illegal parameter is encountered or if a necessary parameter is not supplied.

4) It is not necessary to retype a command if it is already present on the screen. Just position the cursor on the same line, make changes if you wish, and type RETURN. All the normal ATARI editing commands are available.

5) The processing of most commands can be stopped by holding down the START button. This allows you to terminate a long listing, search or single step. Use CTRL-1 to temporarily halt and restart a listing.

DISPLAY MEMORY: D (start addr) (end addr)

This command is used to view data in memory in either hex or character format, depending on the current data format (see TOGGLE). In hex format the data is output to the screen as 1 or more lines of 8 hex bytes separated by spaces. In character format the data is output as 1 or more lines of 24 byte character strings. On each line, the address of the first byte precedes the data.

In either data format the letter 'A' is appended to the start of each line. This in fact represents the ALTER MEMORY command (see the following command description). The effect is that, once you have used 'D' to display part of memory, you can alter any byte(s) by simply positioning the cursor, typing the change(s), and hitting RETURN. You must type RETURN on each line that you alter for the change to take effect. Also, the current data format must match the way the data was represented on the line. One other limitation in the character mode is that a line containing the character representing $9B is not alterable past that character. The OS cannot handle a record with an imbedded $9B. If you wish to alter a line after a $9B character, redisplay the line starting just past it.

To display memory, type 'D' followed by optional start and stop addresses and then RETURN. You can display up to 32K bytes ($8000) with one command. If you omit the stop address, only a single line of data will be printed. If you omit the start address, the next logical line of data will be printed (either the next 8 or 24 bytes of memory, depending on the data format). One last convenience is that once you have used the 'D' command, OMNIMON! will default to that command if you just type RETURN. This allows you to scroll through memory by holding down the RETURN key. This default will remain in effect until one of the other 'persistent' commands (R or X) are used, at which time they will become the default.

TOGGLE DATA FORMAT: T

As mentioned previously, all numerical data is represented in hex. However, when dealing with ASCII text it is more convenient to work in character format. The TOGGLE command (T) is used to switch between hex and character format. It affects three commands: ALTER MEMORY (A), DISPLAY MEMORY (D) and SEARCH MEMORY (S). Other commands are unaffected by the current data format.

To switch data formats type 'T' (RETURN). Upon first entering OMNIMON! the data type defaults to hex.

ALTER MEMORY: A addr byte byte ...

This command is used to change 1 or more contiguous bytes of memory. You can type the change either as hex bytes (separated by spaces) or as ATASCII character strings, depending on the current data format (see TOGGLE). While it is possible to use the 'A' command by itself at any time, it is not recommended. To display the area of memory first with the 'D' command and then to position the cursor and make the change is much safer (see DISPLAY MEMORY). This way you not only verify that the memory at that location is the memory you intended to change, but also that the current data format is compatible with the data you are typing.

To use the ALTER MEMORY command, type 'A' followed by an address. Use a space to separate the address and the data and then start typing the data. If in hex format, type hex bytes delimited by spaces. If in character format, type a continuous character string. Terminate the command with RETURN. At that point the indicated changes will be made. The command line can be as long as you like or until the computer squawks. SQUAWK!

SEARCH MEMORY: S addr byte byte ...

Searching is something that computers do very well and OMNIMON! has a very nice search function that works in either hex or character mode. It will scan memory for any sequence you specify and display it in a manner similar to the DISPLAY MEMORY command every time it is found. This means you can alter any occurrence of that sequence by simply positioning the cursor, typing the change and hitting RETURN (see DISPLAY MEMORY).

To use the SEARCH MEMORY command, type 'S' followed by the address where you would like the search to begin. Then type a space followed by the search sequence. This will be hex bytes separated by spaces in hex mode or a character string in character mode (see TOGGLE). The search will begin when you hit RETURN. The search sequence can be any length up to the limit of the ATARI terminal input buffer. Even though it only takes a few seconds to search all of memory, a search can be aborted by holding down the START button.

PRINTER ON/OFF: P

If you want a hardcopy record of your OMNIMON! session, you can use the 'P' command to cause anything being output to the screen to be echoed to the printer. In character mode, inverse video characters are printed as normal video and unprintable characters are translated to dashes (-). Otherwise, everything on the screen will show up on the printer. There is even a special single step mode (see EXECUTE) that will trace through a program while outputting only to the printer and not to the screen. This is useful for programs that use the screen in modes other than GRAPHICS 0.

The 'P' command is a toggle function. Typing it once will enable output to the printer and typing it again will disable output. If the printer is not turned on or selected, the message 'I/O ERROR' will result. Care should taken if the printer is enabled while reading or writing to the disk (see READ DISK or WRITE TO DISK).

DISK INPUT/OUTPUT

Anyone who owns my disk utility DISKSCAN knows how useful it is to be able to edit raw sector data on a disk. One of my goals in designing OMNIMON! was to incorporate some of the features of DISKSCAN. Imagine, a resident mini-DISKSCAN!

Well, the end result has far exceeded my expectations. With OMNIMON! you can not only read and write individual sectors, but multiple sectors to and from anywhere in memory. And it not only works in sequential mode, but it can also follow sector links. In fact, you can read in an entire DOS file from a disk without even booting up DOS! And the frosting on the cake is that OMNIMON! works equally well in single or double density, a dream come true for the growing number of double density drive owners.

LINK/SEQ MODE & DRIVE #: L (drive#)

When you first enter OMNIMON!, the program assumes that you wish to talk to drive #1 and that the sector mode is sequential. If you wish to address other drives or follow sector links, use the LINK command to put OMNIMON! in the correct mode. The LINK command is actually two commands in one. When used by itself (without a parameter) 'L' means to toggle from sequential to linked mode or vice versa. When followed by a drive # (1-4), 'L' means to switch the drive ID to the specified drive. From that point on, all disk I/O will be directed to that drive.

To toggle between the sequential and linked sector modes, type 'L (RETURN)'.To direct disk I/O to a different drive, type 'L' followed by the drive # and RETURN.

READ DISK: R (sector#) (buffer addr) (# sectors)

The READ DISK command is one of the most powerful, user friendly functions of OMNIMON!. It can be used to read one or more sectors, either sequentially or linked, from any disk drive, single or double density. We

will start out by using the READ DISK command to read one sector at a time. You will find it behaves somewhat differently when operating on more than one sector at a time.

To read a single sector into memory type 'R' followed by the sector # and RETURN. OMNIMON! will assume a buffer address of $6000 unless you specify something different after the sector #. From that point on OMNIMON! will assume that new buffer address for subsequent disk I/O. Once you have the sector in memory you can operate on it with any of the other OMNIMON! commands including DISPLAY, ALTER, SEARCH, DISASSEMBLE, etc. One convenient feature is that, after a READ DISK command, OMNIMON! will assume the buffer address if you use 'D' or 'X' without a start address. (Try typing 'D (RETURN)' after reading a sector into memory).

Now, if you wish to read the sector which logically follows the last sector read into memory, type 'R (RETURN)'. In sequential mode, the next physical sector on the disk will be read. In linked mode, OMNIMON! will reference the sector link of the current sector to determine the next sector to read. In either case, the new sector will be read into memory at the SAME buffer address, overlaying the old sector.

NOTE: IF THE PRINTER IS ENABLED WHILE READING SINGLE SECTORS, THE SECTOR # AND BUFFER ADDRESS MUST BE SPECIFIED EACH TIME. This is because the printer and disk share the SIO DCB (Device Control Block).

Ready for a couple more examples of user friendliness? One is that the 'R' command, like 'D' and 'X', is a 'persistent' command. That means that once you use the 'R' command, OMNIMON! will default to that command if you just type RETURN. This default will remain in effect until one of the other persistent commands are used. What this means is that you can read through an entire file (or disk, if in sequential mode) by reading the first sector and then simply holding down the RETURN key. One other convenience is that OMNIMON! will not read past the end of file if it is in linked mode. Thus, if you were reading through a file as suggested above, simply hold down the RETURN key until 'EOF' is printed. At that point, the last sector of the file is at the buffer address. You are free to add something to the end of the file (perhaps an autorun vector) and then to write the sector back out with the WRITE SECTOR command.

Reading multiple sectors is somewhat different from reading single sectors. For one thing, the sector #, buffer address, and sector count must be specified each time. The other difference is that, instead of consecutive sectors overlaying each other, the buffer address is incremented between sectors so that the disk data fills memory. The exact amount by which the buffer address is incremented depends on the sector mode and the density of the drive. The effect is that in sequential mode all bytes of the sector are preserved, while in linked mode the sector links are overlayed. This is a desirable feature if you want to read an entire DOS file into memory. If you find this discussion confusing, it is recommended that you read my tutorial on ATARI DISK DATA STRUCTURES in the DISKSCAN USER'S GUIDE or the MARCH 1982 issue of COMPUTE! magazine.

An example may be of help. First, put OMNIMON! in character mode with 'T' and sequential mode with 'L'. Now type 'R 169 6000 8'. This will read in the 8 sectors of the directory into the buffer at $6000. Now type 'D' followed by several RETURNs. You will be able to read the names of the files on the disk. Choose the filename of a short file, put OMNIMON! in hex mode with 'T', and use 'D addr' to display the 5 bytes just prior to the filename. The first byte is the status while the next two are the size of the file and the next two are the start sector. Now put the program in linked mode with 'L'. Read in the entire file with 'R (start sector) 6000 (file size)'. Type 'D' and hold down the RETURN key to scroll through the data of the file.

If the file happens to be a BINARY LOAD file, a slight variation of this technique is recommended. Instead of specifying an arbitrary buffer address of $6000, look at the first sector of the file to determine the load address. Then subtract 6 bytes to account for the load vector and use that number as the buffer address. Then, after the file is loaded, it can be executed, searched, disassembled or otherwise manipulated to your hearts content. Notice that the entire program will be loaded to the correct place in memory only if there is but one load vector at the beginning of the file. OMNIMON! as a rule ignores load vectors. If you must load a file with multiple load vectors, use DOS.

Once you have a binary load file in memory you can create a boot disk by switching over to sequential mode and writing the program back out to a disk starting at sector 1. You will need to leave 6 overhead bytes at the beginning of the first sector. See the ATARI OPERATING SYSTEM USER'S MANUAL for details on the boot process. The converse of this process would be to create a binary load file from a boot record. This can be done by first booting up DOS, going to OMNIMON!, and reading in the boot record in sequential mode to a convenient place in memory. Then you would exit back to DOS and do a BINARY SAVE on that portion of memory. Then you may have to use OMNIMON! to change the load vector at the beginning of the file to make it load in at the correct place. In fact, you may have to use OMNIMON! to load the file if it loads on top of DOS. The same technique can be used to make a binary load file out of a cartridge but you will have to append a few load vectors to get the program going. For example, the following 3 load vectors should be appended to the end of BASIC: 6A 00 6A 00 90 E2 02 E3 02 F6 F3 E0 02 E1 02 00 A0.

One final convenience is that each time a single sector is read, its # is printed along with the buffer address. This also occurs when multiple sectors are read, but only when the printer is on. Thus, if you read in an entire file with the printer on, you get a sector map of that file. Now if, while inspecting the file in memory, you find that you wish to make a change to the file on disk, you can compare the buffer address to the sector map of the file to determine the sector where that piece of data resides. Then you can use 'R sec#' to fetch that sector, make the change, and use 'W sec#' to store the sector back on disk.

WRITE TO DISK: W (sector #) (buffer addr) (# sectors)

The WRITE TO DISK command allows you to write one or more sectors worth of memory out to disk. The one big difference between it and the READ DISK command is that it only works in the sequential mode. That means that it will not create a DOS file, i.e., it will create neither a directory entry nor sector links. If you do wish to create a DOS file out of memory it is best to use the BINARY SAVE option of DOS. However, it is not always possible to get DOS into memory without losing your data. In this case, OMNIMON! may be the only way save your data. If you do wish to create a DOS file out of the memory you have written to disk with OMNIMON!, use the technique described in the last section. You could also use the BINARY LOAD FILE function of DISKSCAN in the sequential mode which will pick up sectors and redeposit them as a DOS file.

IMPORTANT: Use a scratch disk when writing multiple sectors worth of memory to disk with OMNIMON!. The program pays no attention to data already on the disk and may overlay it.

The primary purpose of the WRITE TO DISK command is to support the modification of one sector at a time. A typical scenario is as follows:

> Turn the printer on, read a file into memory as described in READ DISK, and turn the printer off. Search the file to find the data to be changed. Compare the address of the data in memory to the sector map created while the file was being read in. Read that particular sector into memory with 'R sec#'. This insures that you not only have the data of the sector but also the sector link. Then alter the sector and write it back out with 'W sec#'.

Another application for the WRITE TO DISK command would be to move a block of memory from one location to another. This is accomplished by writing the data out from one buffer address and reading it back in at another. It is important to use a scratch disk for this operation or at least be very careful that you are writing to an unused portion of a disk.

Be careful when omitting the sector # because the default has already been incremented in anticipation of the next READ DISK. In fact, the only safe time to omit the sector # is when that sector is the last one of a linked file.

SEVERAL WAYS TO ENTER OMNIMON!

We have seen how to enter OMNIMON! by holding down OPTION and pressing SYSTEM RESET. This causes a normal warmstart followed by a jump subroutine (JSR) to OMNIMON!. When you exit OMNIMON! after entering it in this way (by holding down START and pressing RETURN), the warmstart goes to completion in a normal fashion. This is fine for some applications but there is another way to enter OMNIMON! which disturbs the program running as little as possible.

When you hold down SELECT and press SYSTEM RESET, the program running at the time is interrupted. However, instead of doing the entire warmstart, parts of it are skipped over so as to preserve the state of the system as much as possible. Specifically, the OS variables and the stack are left undisturbed. Usually this allows you to reenter the program by simply exiting OMNIMON! in the normal fashion. For instance, you can pop into OMNIMON! from either DOS or BASIC, execute some OMNIMON! commands, and pop back into the interrupted program almost as if you had never left it. I say 'almost' because the OS is likely to return a bogus value if it was waiting for a keystroke when it was interrupted. For that reason it is best to hit BREAK upon return to the program. Of course, if the program makes use of any graphics other than MODE 0, it is unlikely that you will be able to successfully reenter the program without restarting it. This is also true of programs which alter the interrupt RAM vectors ($200-$224) because OMNIMON! restores them to their original values.

There are a couple of small problems with using SELECT/RESET (instead of OPTION/RESET) to interrupt DOS or BASIC. OMNIMON! makes use of the SIO interrupt routines in the OS ROM by altering the interrupt vectors at $20A-$20D. This is so the printer and disk interface of OMNIMON! will work even if DOS is not in memory. Now if you return back to DOS with START/RETURN these interrupt vectors will remain in effect. But DOS hangs up occasionally unless it is using its own special SIO handlers. If you wish DOS to restore its special vectors, exit OMNIMON! with SYSTEM RESET. Another problem with SELECT/RESET is that MEMLO ($2E7) gets restored to $700 so that the FMS or any other program in low memory is unprotected. For that reason it is best to exit OMNIMON! back to BASIC with RESET. Alternatively, always use OPTION/RESET to enter OMNIMON! from BASIC.

Another way to enter OMNIMON! is particularly useful for debugging assembly language programs. This is accomplished by putting 'JSR $C001' at critical points within the program. At each of these points OMNIMON! will be entered and you will have all of its facilities available for examining the intermediate results of your program. When you are ready to continue executing your program, just exit OMNIMON! with START/RETURN. There are some restrictions on this technique however, specifically special graphics and time critical I/O.

Yet another way to enter OMNIMON! is from BASIC with a 'X=USR(49152)'. In fact, this is the recommended way to enter the monitor if you have not modified the interrupt vectors at $FFFA-$FFFD as described in the OMNIMON! INSTALLATION INSTRUCTIONS. You can exit back to BASIC in the usual manner (START/RETURN).

It should also be pointed out that OMNIMON! will be entered automatically if a 6502 BRK instruction (0) is ever executed. Thus, you can set a breakpoint anywhere in your program by storing a 0. When you pop into OMNIMON! after executing a BRK instruction, you should restore the original instruction and subtract 2 from the PC. Now you can continue

executing your code when you exit OMNIMON!.

CPU REGISTERS: C

You will notice that, upon entering OMNIMON!, the 6502's internal registers are printed out with the following heading:

PC NV-BDIZC ACCUM X-REG Y-REG STACK

The meanings of these headings are self-explanatory except for 'NV-BDIZC'. These are the individual bits of the status register spelled out. Thus, this is a snapshot of the state of the CPU just prior to entering OMNIMON!. The PC (program counter) is pointing to the next instruction to be executed. The program will continue executing at this point when you leave OMNIMON! with START/RETURN.

The CPU state can be examined at any time with the CPU REGISTERS command 'C'. In addition, the CPU state can be changed by simply positioning the cursor over the value, typing the change, and hitting RETURN. The new values for the registers will be in effect when you leave OMNIMON! to resume execution of the suspended program. The only CPU register that cannot be changed directly is the stack pointer. This can be changed only by the PUSH STACK (+) and POP STACK (-) commands.

One application for the 'C' command is to GOTO anyplace in memory. This is accomplished by altering the PC to point to the address where you wish execution to resume when you press START/RETURN. Typically this might be back to DOS, whose address can usually be found by looking in location $000A (DOSVEC).

Another area of interest is the stack. Remember, the stack pointer always points to the next FREE entry. All the values between the stack pointer and $1FF are typically return addresses of nested subroutine calls. This, in fact, is a vertical cross section of the execution history of the program. This is extremely useful for finding your way around in a program you wish to modify in some way. If you wish to to locate the part of a program which is performing a certain function, just start the program executing that function and press SELECT/RESET. Because the stack is preserved with this method of entering OMNIMON!, you can tell where the program is and where it has been by noting the PC and the return addresses on the stack. Another way of locating a certain piece of code is to search ('S') for a particular address it might reference.

PUSH STACK: + byte byte ...

The PUSH STACK command is for adding bytes to the stack and thereby increasing the stack pointer (which grows downward in the 6502). These bytes will be available to the code pointed to by the PC when OMNIMON! is exited. Notice that the first byte after '+' is the first one to be pushed onto the stack.

Please note that the stack pointer displayed with the 'C' command is not the ACTUAL stack pointer while OMNIMON! is running. OMNIMON! uses the stack for its own purposes and is actually nested somewhat deeper. It is not wise to make changes directly to the stack unless you use PUSH STACK or POP STACK. Even then you must be careful not to cause the stack to overflow or underflow.

POP STACK: -

The POP STACK command takes bytes off of the stack one at a time and decreases the stack pointer (which actually increases in value). Be careful to not cause the stack to underflow.

DISASSEMBLE MEMORY: X (start addr) (stop addr)

Just as it is possible to display memory in hex or character format, it is also possible to translate 6502 machine code to assembly language. OMNIMON! does this in a handy fashion by printing out the object code along with the instruction. Once again, it is possible to change the object code (to the left of '*') by positioning the cursor, typing the change, and hitting RETURN. Another convenience is that the value at the address specified with indirect addressing modes (without regard to the index register) is printed in parentheses.

Just like the 'R' and 'D' commands, 'X' is 'persistent'. Once you have disassembled one or more instructions, you can continue disassembling simply by holding down RETURN. This will remain in effect until 'R' or 'D' are used. The disassembler can be aborted at any time by pressing the START button.

EXECUTE MEMORY: E (option/# steps)

The EXECUTE MEMORY command is actually a single step command in disguise ('S' is used for SEARCH). This command causes the instruction pointed to by the PC to be executed. Then the registers are printed out along with the NEXT instruction to be executed. If the step count was 1 (or not specified) then execution will stop. Otherwise it will continue single stepping through the code for the specified # steps. The maximum number of steps at one time is 31 for reasons soon to become clear.

While the low order 5 bits of the optional parameter are a step count, the high order 3 bits have special meaning. The MSB means 'step forever'. Thus, 'E 80' means 'step forever and print the trace to the screen'. Notice that the trace will also be echoed to the printer if it is enabled. Stepping can be aborted by pressing START.

Bit 6 of the parameter means 'don't print the trace to the screen'. However, the trace will still be output to the printer if it is enabled. Thus, 'E C0' would step forever without printing the trace to the screen. In combination with the printer this is useful for stepping through programs which use special graphics modes.

Bit 5 of the parameter means 'sample the results of every 32 instructions'. Thus, 'E E0' would step forever without printing to the screen and the trace would be output to the printer every 32nd instruction (if it is enabled). This is kind of a weird mode, but somebody may find a use for it someday.

One other nice feature of the 'E' command is that it will treat a call to the OS as a single instruction instead of stepping through all the code in the OS. OMNIMON! does this by temporarily giving up control of the CPU but intercepting it on the return from the OS. However, you should avoid stepping through CIO calls to the screen editor (E:) unless printing to the screen is disabled with bit 6. OMNIMON! considers any address above $C000 to be OS.

We have seen that the EXECUTE MEMORY command is very powerful and flexible. One restriction, however, is that it will not step through a 'SEI' instruction. If you are stepping through a program and encounter a SEI, disassemble on past it to find the 'CLI'. Just past the CLI put a temporary BRK instruction (0). Now step through the SEI. OMNIMON! will temporarily lose control of the program but will regain it when the BRK instruction is executed. Now restore the original value to the location where the BRK was set. After subtracting 2 from the PC you are ready to continue stepping.

JSR: J addr

The JSR command is a very powerful feature for executing a subroutine and returning control back to OMNIMON!. It can be used for testing out subroutines during the development of an assembly language program. With some care it can also be used to call the OS to, say, format a disk.

When you execute the 'J' command you will notice that the registers are printed out but that the subroutine is not yet executed. In fact, the 'J' command does nothing more than change the PC to the specified address and push the address of OMNIMON! on the stack to act as the return address for the subroutine. Now you are free to set up for the subroutine call by altering the registers or memory if necessary. When you are ready to actually execute the subroutine press START/RETURN. Upon return you will notice that the PC is restored to its original value but that the other registers reflect the results of the subroutine.

As an example, put a fresh disk (or one you don't mind formatting) in drive 1. With the printer disabled, store a 1 in $301, a $21 in $302, and a $80 in $303. Now execute a 'J E453' and press START/RETURN. The disk in drive 1 will be formatted and then control will be returned to OMNIMON!.

Sometimes after interrupting a program with OMNIMON!, you will not be able to restart it without reinitializing it. The start and initialization addresses for a program are typically at $000A and $000C respectively. Since a proper initialization routine is always a subroutine, you can use 'J (init addr)' to initialize the program. When control returns to OMNIMON!, you need only change the PC to the start address and exit OMNIMON! with START/RETURN to restart the program.

QUESTIONS?

I welcome comments, suggestions and dealer inquiries:

DAVID YOUNG
CDY CONSULTING
421 HANBEE
RICHARDSON, TX 75080
(214)235-2146

LIMITED WARRANTY

For a period of one year following the date of purchase CDY CONSULTING will repair or replace any OMNIMON! unit proven to be defective. Please return the defective unit to the place of purchase.

Binary Load Files

This is one area that most people are a little fuzzy on. It's not surprising really, since there does not appear to be any definitive documentation on it anywhere! An exhaustive presentation will be made here even though it will be quite short.

Definition: **load vector** - from 4 to 6 bytes consisting of 2 optional bytes of FF FF, a 2 byte start address, and a 2 byte end address (in that order).

Example: FF FF ØØ Ø6 Ø2 Ø6 - The first 2 bytes are optional and ignored during the load process. The second 2 bytes are a start address of $6ØØ and the last 2 bytes are an end address of $6Ø2.

The only time that the first 2 bytes of FF FF are required is at the beginning of a binary load file. If those bytes are not there, DOS will refuse to perform the binary load. (The 'G' command of OMNIMONL will, however, load it gladly.) The rest of the time the first 2 bytes of FF FF, if they exist, are ignored. The only time that these 2 optional bytes should occur anywhere else but at the beginning of a file is if 2 binary load files were appended together.

What does a load vector do? It tells DOS (or the 'G' command of OMNIMON) where in memory to put the 1 or more bytes which follow in the file. How many bytes is determined by subtracting the start address from the end address and adding 1. In the example above, 3 bytes would be read from the file and put in locations $6ØØ to $6Ø2.

What happens when enough bytes have been read in to satisfy a load vector? These things will happen in this order:

1) Locations $2E2 and $2E3 will be examined. If they are both zero, goto step 2. If they are nonzero, a JSR will be made to the address contained in these locations. Upon return, zero $2E2 and $2E3 and fall into step 2.
2) If the end of file is not reached (i.e., there are more bytes in the file), another load vector is assumed to immediately follow and will be processed as previously described.
3) If the end of file (EOF) is reached, examine locations $2EØ and $2E1. If they are zero, terminate the binary load. If they are nonzero, do a JSR to the address in these locations. Upon return, terminate the load.

Still confused? Let me try to simplify. If you see a load vector like 'E2 Ø2 E3 Ø2', you know that the subroutine at the address specified in the following 2 bytes will get executed immediately, prior to continuing the load process. If you see a load vector like 'EØ Ø2 E1 Ø2', you know that the subroutine at the address in the following to 2 bytes will be executed after the end of file is reached.

Now that you understand everything there is to know about binary load files, let's take a typical example: converting the BASIC cartridge to binary load file. I use this example because it is instructive and, because of the lack of copyright notice, appears to be legal. Using this technique on other cartridges could be illegal and may not work anyway due to the booby traps designed to prevent them from running out of RAM.

Whenever attempting something new, it is advisable to try it out manually from OMNIMON first whenever possible. Let's see what it takes to get the BASIC program to run out of RAM:

-Turn on the computer with BASIC installed and pop into OMNIMON.
-Insert a formatted scratch disk into the drive and execute the following command: 'W1 A000 40 (RETURN)'.
-Remove the BASIC cartridge, boot up DOS and pop into OMNIMON.
-Because OMNIMON restores MEMLO to $700, we should reinitialize DOS by doing a JSR to the address at DOSINI ($C,D). For 2.0S that would be 'J1540 (RETURN) (START/RETURN)'.
-Move the screen down by storing a $90 in location $6A and doing a JSR $F3F6: 'A 6A 90 (RETURN)','J F3F6 (RETURN) (START/RETURN)'
-Read BASIC back into memory with 'R1 A000 40 (RETURN)'.
-Find the initialization address by looking at location $BFFE. Since that is $BFF9, execute 'J BFF9 (RETURN) (START/RETURN)'.
-Find the start address by looking at location $BFFA. Since that is $A000, execute 'J A000 (RETURN) (START/RETURN)'.

BASIC will now come up running. Now we want to create a binary load file to simulate the last 4 steps:

-Type 'DOS' to get to the DOS menu.
-Use the 'K' command to save BASIC as a binary load file giving it the start address of A000 and the end address of BFFF.
-Pop into OMNIMON and put it in linked mode. Read the first sector of the new file into $6000 (see top of page 7 if you don't know how to find the first sector of a file).
-Hold down RETURN until 'EOF' is printed out. You now have the last sector of the file in memory.
-Determine the last byte of that sector in use by looking at the byte count ($607F). Start adding the following bytes at location $6000 + byte count (we are appending to the file):6A 00 6A 00 90 E2 02 E3 02 F6 F3 00 98 08 98 20 06 98 6C FA BF 6C FE BF E0 02 E1 02 00 98.
-Increase the byte count ($607F) by $1E and write that sector back out to the disk with 'W (RETURN)'.

Here is an explanation of these load vectors: The first 11 bytes move the screen down and the rest load a little routine into $9800 to initialize and start the cartridge. Disassemble them to see how. Except for the booby traps, you should be able to easily extend this technique to 16K cartridges. However, some cartridges assume that memory is clear, so you can append the following load vectors: 09 98 23 98 A0 00 A9 07 85 D5 A9 00 85 D4 A9 00 91 D4 E6 D4 D0 F8 E6 D5 A9 30 C5 D5 D0 F0 E2 02 E3 02 09 98 60. This will clear out DOS before starting the cartridge.

OMNIMON! INSTALLATION INSTRUCTIONS

NOTE: If you purchased OMNIMON! installed in the OS piggy-back board, ignore this sheet. Instead, refer to the installation instructions provided with the board.

The OMNIMON! program comes in a 2732 compatible ROM. This can be installed only in special hardware such as RAMROD MMOS by NEWELL INDUSTRIES which accepts a 2732 in the $C000 block of memory. Follow the instructions provided with this hardware for installation of the chip.

To make sure the installation is correct up to this point, power up the computer. If everything appears to be normal, try entering OMNIMON! from BASIC with a 'X=USR(49152)'. If it fails this test then double check the installation of the chip. If this works properly then proceed with the next paragraph.

For proper operation of OMNIMON! the following locations should be changed in the OS (assuming your OS is in EPROMS):

$FFFA-$FFFD - $CE,$CF,$F5,$CF

This allows the SELECT and OPTION keys to work in conjunction with SYSTEM RESET for entering OMNIMON!. If If the OS chips were provided with the RAMROD then these changes have already been made. If you do not make this mod then it is possible to enter OMNIMON! from BASIC with 'X=USR(49152)'. However, this is less convenient than the SELECT/RESET or OPTION/RESET technique, which interrupts any program which may be running.

If the system fails to power up after you have made these mods then the mods were not done correctly. When done correctly the system should power up normally and you should be able to enter OMNIMON! with SELECT/RESET or OPTION/RESET. Installation of OMNIMON! is now complete.

OMNIMONA!
For Advanced Users

By deleting the HELP command and shortening the user messages, room was freed up for a few more desirable commands and useful features. This document describes these additional features. Please refer to the OMNIMON! USER'S MANUAL for a description of the standard version of the monitor. A copy of OMNIMONA! can be acquired by sending a 350nS 2732 EPROM + $2.00 postage or $10.00 to CDY Consulting, 421 Hanbee, Richardson TX 75080. OMNIMONA! can be freely copied for dissemination among the community of OMNIMON! users.

1) Hex converter: H #

   # can be either in hex or decimal. Decimal #'s are terminated with a non-hex character. For example, 'H100T' yields '$64 = 100'.

2) Verify memory: V addr#1 addr#2 #bytes

   Compare the specified # bytes starting at the two blocks of memory specified by addr#1 and addr#2. Print the differences.

3) Happy upload and download:

   If the disk is read or written to with a sector # of $800 or greater, the Happy drive treats the sector # as an internal buffer address ($800-$13FF). On multiple reads or writes the sector # is incremented by $80 each sector. This is not only useful for programming the Happy drive but is also handy for storing stuff on a temporary basis.

4) RESET mod:

   There is support for the hardware mod which hooks the SYSTEM RESET switch into the RESET line of the CPU. This is accomplished by inserting a 47 ohm resistor into the pads provided on the motherboard near CR103 and R183. A momentary switch to connect the resistor is also needed unless you wish to do a coldstart everytime SYSTEM RESET is pressed. With this mod and OMNIMONA! it is possible to recover from a system lockup. This is accomplished by pressing the momentary switch and pressing SELECT/RESET. This will pop into the monitor with the PC reflecting the instruction the CPU was trying to execute when it locked up (probably an illegal one). At that point RESET can be pushed by itself to do a normal warmstart.

5) P command enhancement:

   The trace turned on by the P command can now be redirected to any output device. If it is desired to output to something other than the printer, store the device specification someplace in memory and point $105 to that location. The P command will then open up an I/O channel to that device and the next P command will close it. For example, if you were to store 'D:TEMP ' (notice the blank used as a terminator) at $600 and store 00 06 at $105, the P command would open up a disk file. This feature is especially useful in conjunction with the disassembler or the single-step.

OMNIMON! UPDATE: ZAXXON PATCH

The growing number of enthusiastic OMNIMON'ers I hear from everyday give credence to my belief that OMNIMON! is fast becoming the de facto monitor for the ATARI 400/800. It joins a growing number of products which result in an enhanced operating system (e.g., FASTCHIP, RAMROD, MPP-1100, etc).

Now, what if I were to tell you that there is a well known, respectable software publisher that is deliberately trying to be incompatible with all users except those with standard ATARI OS hardware? Who (except maybe ATARI) would be so silly as to pull a stunt like that? Yes folks, it seems that Datasoft Inc. has decided to publish the new disk version of ZAXXON with a protection mechanism so clever that it runs only with standard ATARI, plain vanilla OSA or OSB with the sloth-like ATARI floating point chip. Evidently Datasoft also thought it would be cute to keep the unsuspecting buyer unaware of his predicament until he got home and discovered that his proud new purchase would not run on his machine until he ripped out all of his expensive non-ATARI OS hardware, to which he has become very attached. Well, perhaps the person in charge of coming up with a protection scheme (that must be quite fun really!) decided that it would heighten the player's anticipation if it were a real challenge to get it to run on the system in the first place. Now that would be a novel idea! Promote a new game with a media blitz for months prior to release and then incorporate a protection scheme so clever, so delightfully devious that it will not run on ANYONE's system, even if all the equipment (down to the 825 printer and 830 modem) were ATARI. The initial challenge of this ingenious game would be to break it so that it would run at all!

But let's get to the matter at hand: how to get ZAXXON to run on our enhanced OS's. It turns out that there is a very simple 3 byte patch to the ZAXXON disk which does the trick:

Sector $17D - bytes $26-$28
From: 23 C0 1A
To : 43 EB 3A

In OMNIMONese this translates to:

```
R 17D 6000
A 6026 43 C0 3A
W 17D 6000
```

So this should keep everyone satisfied unless Datasoft decides to keep us on our toes by shuffling their protection around every so often. And what about other programs which might come out with similar, if just as inane, protection schemes? If someone will make me aware of it, I will enjoy coming up with a patch to make it run.

David Young
CDY Consulting
(214)235-2146

OMNIVIEW 80 Columns!

OMNIVIEW XL/XE (standard XL/XE computers)

OMNIVIEW 256 (upgraded 256K XL/XE computers)

OmniWriter 80 by C. David Young **OMNIVIEW** CDY Consulting (214)235-2146 07/01/86

If you have never done 80 column word processing, you may not know what you are missing, but believe me, once you try it, you will never go back to 40 columns! You too can enjoy professional quality word processing on your ATARI 8 bit computer by installing an OMNIVIEW 80 columns enhancement. Check out these valuable features:

Environments. Letter/Data Perfect, Atariwriter Plus (130XE version), BASIC, DOS, MAC65, and many programs which use standard E: device. In addition, OmniWriter (a full feature 80 column word processor and text editor) and OmniTerm (an 80 column communication program with XMODEM, capture, macros, etc.) are provided with every OMNIVIEW sold. Also, a VT100 emulator with built-in Kermit is available for $10.00. To get your Atariwriter Plus converted to 80 columns, send a copy of the 130XE side (or the original plus a blank disk) to us with $10.00.

Features. Besides crisp, legible 80 columns output, OMNIVIEW offers many other features. OMNIVIEW for the 400/800 has resident ramdisk handlers to support AXLON compatible ram upgrades up to 1 MB! Likewise, the OMNIVIEWs for the XL/XE computers have resident ramdisk handlers for XE compatible ram upgrades. Plus, the latter has its own 800 compatible OS with built in translator disk, reverse OPTION/BASIC selection, coldstart from the keyboard (reboot without losing contents of ramdisk), FASTCHIP floating point, and built-in 80 column ATRMON. 400/800 is plug-in. XL/XE will require soldering if OS is unsocketed. A non-composite monitor is recommended for serious 80 column work.

OMNIVIEW 400/800 (400/800 computers)

Simulated 80 column screen. OMNIVIEW's characters look a little different but are very clear and legible.

Other Products for Your 8 Bit ATARI!

Ramrod XL/XE (for 130XE and 800XL):

This is a small board with 3 sockets and a switch that is mounted externally. It plugs in place of the OS chip to allow up to 3 operating systems to be resident. Use it to retain the original OS when installing OMNIVIEW. OMNIMON XL is also an option.

256KXL (for 800XL and 1200XL):

This is the very best 256K upgrade for these two computers because it is more 130XE compatible than any other on the market. A great compliment to OMNIVIEW, it can be used as a ramdisk or as an extension of your text buffer with OmniWriter or to run the 130XE version of AtariWriter Plus. Can be ordered with or without RAM.

128K RAM upgrade (for old 800):

For all of you proud 800 owners we sell an AXLON compatible 128K board. Use it with all programs designed to work with AXLON (SYNFILE/CALC, HAPPY, etc.). We also provide a patch to make the 130XE version of AtariWriter Plus work! Other uses for this RAM are as a ramdisk or as an extenstion of your text buffer in OmniWriter.

To order or for more information contact:

CDY Consulting
421 Hanbee
Richardson, TX 75080
(214)235-2146

Introduction to OMNIVIEW

Congratulations on purchasing the most powerful 80 column system for the ATARI 8 bit computers! As of this writing there are a variety of programs that will work 80 columns with OMNIVIEW:

Word processors: **OmniWriter** is the one provided free with OMNIVIEW. We consider it overall the most powerful word processor for the ATARI. We are constantly improving it and upgrades are always available for $10.00.

AtariWriter Plus (130 XE version) will work on 130XE compatible computers. Send a copy of the disk (or the original plus a blank disk) to us with $10.00 for conversion to 80 columns.

Letter Perfect can be converted by you using the patches provided in this manual or we will do it for you if you will send us a copy of the disk with $10.00.

Data base: **Data Perfect** can be converted by you using the patches provided in this manual or we will do it for you if you will send us a copy of the disk with $10.00.

Communications: **OmniTerm** is provided free with OMNIVIEW and is useful for talking to a BBS.

VT100 emulator with built-in Kermit is useful for talking to mainframes and general communication tasks. It is available from us for $10.00.

Others: **Many others** that use the standard ATARI screen editor (DOS, Basic, MAC65, etc.) will work by simply turning on 80 columns while running the program.

Getting OMNIVIEW Installed

Before using OMNIVIEW we must get it installed in you computer. How difficult this will be depends on your computer and your proficiency with electronics. Almost anyone can do the installation in the 800. The installation in the 800XL is also not difficult if your machine is socketed. Installation in an 800XL without sockets or in a 130XE should only be attempted by a skilled technician. Follow the installation instructions for your model and then skip to the section on hooking up your monitor.

400/800 Installation Instructions

The OMNIVIEW for the 400/800 will plug onto either the OMNIMON piggyback board (on the 400 or 800) or onto Ramrod OS board (on the 800). Follow the instructions for installating whichever board you have and make sure the board works before plugging in OMNIVIEW! (However, if you purchased it with the Ramrod OS board it may already be plugged in.) Power up the system and make sure it acts normally. You might even try popping into OMNIMON, if present, by holding down OPTION and pressing RESET. Once you are confident that the board is working correctly then you can proceed to plug in the OMNIVIEW.

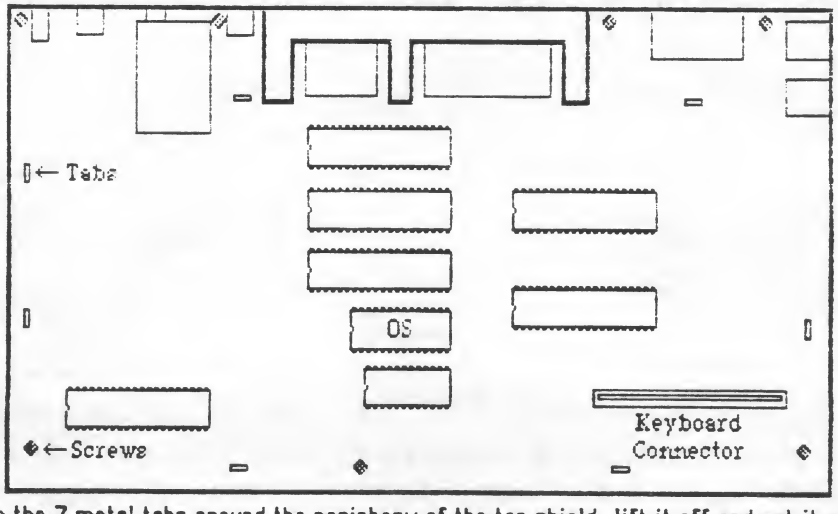
On the piggyback board you will replace the chip with the OMNIMON label with your OMNIVIEW chip. First note the orientation of the notch on the right end of the OMNIMON chip. You must plug the OMNIVIEW chip in the same orientation or you will burn it up! Carefully unplug the OMNIMON chip by inserting a flat blade screwdriver under the chip and gently rocking it back and forth until it is free. Now plug the OMNIVIEW chip in the socket being careful not to bend any pins. To get the pins to line up with the holes you may want to press each side of the chip against a flat surface to bend the legs in slightly.

On the Ramrod OS board the OMNIVIEW chip can be plugged into either socket Z9 or, if there is an OMNIMON in Z9, into Z10. However, if you plug it into Z10 then the board must be modified to add a toggle switch to select either Z9 or Z10. Follow the instructions under RAMROD UPDATES step 1B to install the switch. The notch in the chip should be toward the top of the board.

130XE Installation Instructions

Caution: This installation should be attempted only by a skilled technician! A chip must be desoldered which can lead to distruction of the board if not properly done. If only ATARI had used a socket for the OS chip!

1) Turn the computer upside down and loosen the 4 crosspoint screws holding the case together. Carefully turn the computer over and collect the screws as they fall out.
2) Lift off the top of the case and set it aside. Carefully unplug the keyboard and set it aside.



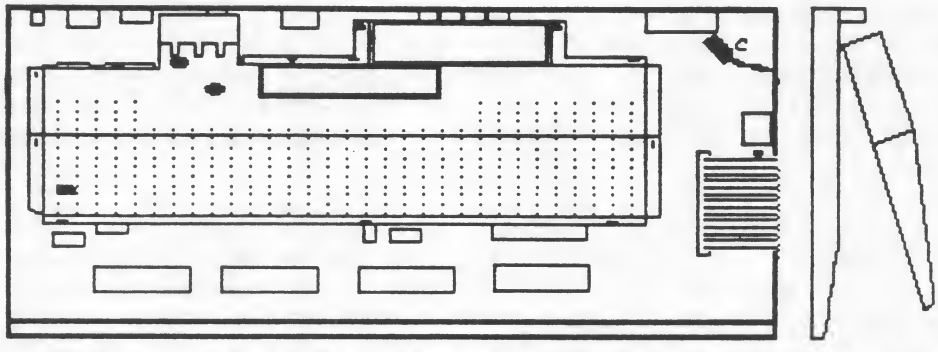
3) Straighten the 7 metal tabs around the periphery of the top shield, lift it off and set it aside.
4) Remove the 7 screws around the periphery of the motherboard and lift it out of the case.
5) The bottom shield can now easily be removed from the bottom of the motherboard. Set it aside.
6) Referring to the diagram, locate the 28 pin OS chip. Unsolder the chip. Nobody should attempt to do this unless they have a lot of soldering experience.
7) Solder a 28 pin socket in place of the OS chip and, noting the orientation of the notch, plug the OS chip back in. Test the computer by plugging the power and monitor cables back in and turning the computer on. If the screen comes up in BASIC then the socket installation was successful. Unplug the ATARI OS and plug OMNIVIEW XL/XE in its place. Optionally, a Ramrod XL/XE can be plugged into the socket and then both the OMNIVIEW XL/XE and the original OS plugged into the Ramrod XL/XE.
8) Complete the installation by reversing the disassembly instructions.

OMNIVIEW-XL Installation Instructions

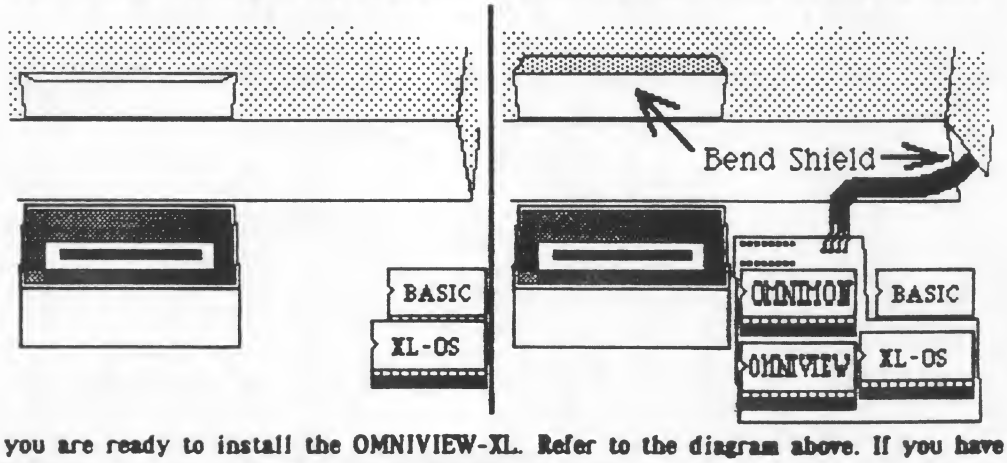
Tools Required: Crosspoint screwdriver, flat blade screwdriver

1) Turn the computer upside down and remove the six crosspoint screws holding the case together.
2) Turn the computer upright and lift the top half of the case from the left, pivoting on the right edge, and lay it upside down to the right of the bottom half.
3) Now we wish to gain access to the area underneath the metal shield. If your computer has a single screw and tabs around the edge holding the shield down, go to 3A. If there are screws (with nuts) holding the shield down, go to 3B.

   3A) Simply remove the screw and straighten the tabs so that you can lift the shield from the front, pivoting about 30 degrees on the remaining two screws at the back. This will bend the two metal tabs at the back slightly but this is of little consequence (see diagram below). Go to step 4.



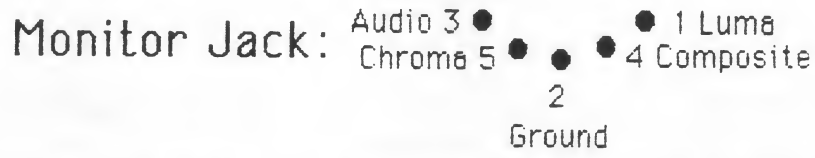
   3B) You will need to remove the motherboard from the bottom half of the case. In this case you will probably want to disconnect the keyboard cable by gently pulling it out of the connector on the motherboard. Remove the remaining screws holding the motherboard to the case (top right and left corners and between the joystick ports) and remove it by lifting from the left side and prying the case around the joystick ports on the right. It is a tight fit but it should pop out. Once the motherboard is free, you can remove the nuts and screws holding the shield to the motherboard. It is recommended that you leave the two at the back on either side of the expansion port. In this way you can lift the shield from the front to about a 30 degree angle, bending the back tabs slightly (see the diagram above).



4) Now you are ready to install the OMNIVIEW-XL. Refer to the diagram above. If you have a RAMROD-XL, go to step 5. Else, locate the XL-OS chip, a 28 pin chip about 2 inches to the right of the cartridge slot. Remove it by inserting the flat screwdriver between the chip and the socket and gently prying and rotating the screwdriver. If the OS chip is soldered directly to the board (this should rarely be the case), you will need to have a skilled technician remove the chip and install a socket.
5) Insert the OMNIVIEW-XL chip into the empty socket on the motherboard or on the RAMROD-XL. **Make careful note of the orientation of the chip, otherwise you may burn it up!**
6) Complete the installation by reversing the disassembly instructions.

Hooking up your monitor

Some people are satisfied using OMNIVIEW with a composite color monitor or even a TV set, especially if the color is turned off. However, the best 80 column output is achieved with either a monochrome monitor or a color monitor with separate chrominance and luminance inputs (like the Teknika MJ10, Commodore 1702 or 1802, etc.). **But even if you have the right monitor, the output will not look good unless it is hooked up correctly!** The ATARI monitor jack has three video outputs: composite, chroma, and luma. If you have a monochrome monitor then you must use the luma output. If you have a color monitor with separate chroma and luma inputs then you must use these two outputs from the ATARI.



Depending on your cable, you may need to modify the connector that plugs into the computer or, if your cable has multiple plugs, select the proper plug at the monitor. When the monitor is hooked up correctly each pixel should be distinct and of equal intensity.

Using OMNIVIEW

If you have installed everything, your computer is functioning normally, and your monitor is hooked up correctly, then you are ready to start using OMNIVIEW. If you only care about using one of the programs described in the introduction, like OmniWriter, just load up the program and go. It will turn on the 80 columns automatically. If you want to use 80 columns in other programs that normally use 40 columns, or if you want to use some of the other features, like the ramdisk handlers, then you must read on. Read the next section if you have an 800XL or 130XE. Otherwise, skip to the section on 400/800 OMNIVIEW features.

OMNIVIEW XL/XE/256 Features

Turning on 80 columns:

80 column emulation is activated from the keyboard by typing CTRL-A and hitting RESET. To return to 40 columns, type a key without CTRL and hit RESET. Don't try this if running the OS in RAM. This technique should work with BASIC, DOS, and most programs that behave nicely when you hit RESET. Other ways to turn on 80 columns are 'X=USR(49152)' from BASIC or 'JSR $C001' from assembly language.

Installing the resident ramdisk handlers:

The resident Ramdisk handlers in OMNIVIEW XL/XE/256 allow you to use

the extra 64K RAM of the 130XE or the extra 192K RAM of the 800XL with the 256KXL upgrade as an ultra fast single density disk drive in conjuction with any DOS which uses standard SIO calls ($E459 and $E453) and does not hide itself underneath the cartridge or OS (e.g. ATARI 2.0S, MYDOS, SMARTDOS, etc.). In addition you will find it possible to use the Ramdisk with boot programs like Letter Perfect and Data Perfect.

The easiest method to activate the ramdisk is to load (with the binary load function of DOS) one the files on the OmniWriter disk called INSTALL1 through INSTALL4. These will install the ramdisk as drive 1 to 4 respectively. You could rename one of the files to AUTORUN.SYS if you want the ramdisk installed automatically on bootup. Once the ramdisk is installed it must first be formatted (as you would any new disk) before it can be used. Also, if it is to be used as drive 1 then the DOS files must be written to it.

Another method is as follows: Type a number (1 to 8) corresponding to the drive number you wish to assign the Ramdisk, hold down the START key, and press RESET. If you do not hit a number prior pressing START/RESET, drive 1 will be assumed. In Letter/Data Perfect this combination is also used to change the screen colors, so assign the Ramdisk as drive 3 if you do not wish to use it in these environments.

For example, in BASIC:
1) Type DOS to go to DOS. Now type 2 and START/RESET to install the Ramdisk as drive 2.
2) Since you are now back in BASIC, go to DOS again and format and write DOS files to drive 2.
3) Now type 1 and START/RESET to install the Ramdisk as drive 1 if you so desire.

Or from assembly language:

```
LDA #2 (drive #)
STA $94
LDA $D301
AND #$7F
STA $D301
JSR $CFAE
LDA $D301
ORA #$80
STA $D301
RTS
```

On a 130XE with OMNIVIEW XL/XE there is only enough room for a 512 sector ramdisk. An attempt to use more than 512 sectors of the Ramdisk will result in an I/O ERROR. If you have OMNIVIEW 256 in a machine with at least 256K of memory then you can have up to 1512 sectors in your ramdisk, depending on the DOS. If you want more than 720 sectors then you must tell your DOS that the ramdisk is double sided. The ramdisk will not work double density.

400/800 compatibility:
This feature ameliorates the worst problem associated with the 800XL and130XE, namely that they won't run so much of the older ATARI software! It does this by having an ultra compatible 400/800 style OS which will copy itself into RAM, freeing up the $C000 page. There are still some highly protected games which, as a part of their misguided protection schemes (e.g., Electronic Arts), refuse to run if your machine's OS is enhanced in any way. That is their problem. Don't call us about their unethical practices. Complain instead to the publisher of the program.

To copy the OS into RAM (from $D800 to $FFFF), hold down the SELECT key while pressing RESET. To restore the OS to ROM, press RESET by itself. From this point on, the RAM version of the OS will be preserved, even if you switch the OS to ROM and back to RAM. Thus, any changes you may make to the OS in RAM remain in effect as long as you do not power down. In addition, if you hold down the SELECT key during powerup, the OS will be copied into RAM and it will stay in RAM even if you press RESET. Please note that the 80 column emulation is not available when running the OS out of RAM.

Basic Activation with OPTION:
The meaning of the OPTION key during powerup is just opposite of the original OS: hold down the OPTION key to activate BASIC. This seems to be the preference of most people.

Scroll control (CTRL-1):
One other convenience is that CTRL-1 has been replaced by the HELP key. Press the HELP key once to stop scolling and again to start scrolling.

Coldstart from the keyboard:
Press HELP and hit RESET. This is the same as powering up except that the contents of the ramdisk are preserved. It is also healthier for your computer than cycling power. However, watch out because HELP is also used to control scrolling. If you have just controlled scrolling with HELP and you want to do a warmstart, be sure to hit some other key before you hit RESET.

Changing screen colors:
It is possible to switch the screen colors in the 80 column mode by holding down the START key while typing a letter. If this does not work (as in Letter Perfect), try holding down the START key while pressing RESET. However, since this combination is also used to install the Ramdisk handlers, read the section on the ramdisk before using this second technique.

OMNIVIEW 400/800 Features

Turning on 80 columns:

If you are not using one of the programs that activates 80 columns automatically, you can do so from the keyboard by holding down the START and SELECT switches and then very briefly pressing the OPTION switch. Then press the BREAK key to clear out the line buffer. It is important that you hold down the OPTION switch as briefly as possible (more of a tap actually). This is because these three switches are monitored during the vertical blank interrupt (VBI). If the VBI detects the closure of all three switchs, it does a JSR $C001 to initialize OMNIVIEW. Holding the switches down longer than one VBI causes the VBI to be reentered, pushing more stuff on the stack and eventually causing the stack to overflow. A great way to lock up your computer! Other ways to turn on 80 columns are 'X=USR(49152)' from BASIC or 'JSR $C001' from assembly language.

Installing the ramdisk handlers:

If you have an AXLON compatible 128K board in your system then you can use the ramdisk handlers in OMNIVIEW to force almost any DOS to recognize it as a single density ramdisk.

The easiest way to activate the ramdisk is as follows:
1) Go into a DOS that uses the ATARI screen editor (that is, the cursor editing controls are active). If you are using a DOS that responds to a single keystroke you must get into a mode where you are entering a line of text, like when typing a filespec during the directory command.
2) Hold down the START key and type control-, (control-comma). That will install the ramdisk as drive 1.
3) If you wish the ramdisk to be something other than drive 1, type the drive #, put the cursor back on the number, and then type control-,.
4) Hit the BREAK key to abort the DOS command.
5) Format the ramdisk with the 'I' command of DOS.
6) If the ramdisk is drive 1 then you must also write DOS files to it with the 'H' command of DOS.

An even easier way if you also have the 8K OMNIMON in the system is to use it to install the handlers. Then switch back to OMNIVIEW and the ramdisk will stay active. The installation can also be accomplished from assembly language by storing the drive # in location $94 and doing a JSR $CF24.

Reversing the screen colors:

You can reverse the screen colors by holding down START and hitting RESET. Alternatively, you can hold down START during the entire boot process. However, you must wait until after the boot process has started before pressing it, otherwise the OS will try to boot the cassette recorder.

Technical Details about OMNIVIEW

OMNIVIEW uses ANTIC mode F (BASIC GR. 8), which gives you a resolution of 320 by 192 pixels. If you use a 4 by 8 character cell, this gives you exactly 80 columns by 24 rows. One drawback to this scheme is that it uses $1E00 bytes (almost 8K) of memory for the screen data. Here is a memory map of the screen data:

```
RAMTOP*256           -> RAMTOP holds the number of pages of RAM
RAMTOP*256 - $126    -> Unused
RAMTOP*256 - $1F0    -> Biginning of display list (after screen data!)
RAMTOP*256 - $1FF0   -> Beginning of screen data (SAVMSC)
RAMTOP*256 - $2001   -> Last byte of free RAM (MEMTOP)
```

Another drawback is that the format of the screen data is not nearly so convenient as BASIC GR. 0 (which is essentially stored as ATASCII). Each character must be translated to pixel data represented by bits in noncontiguous bytes in screen memory. Fortunately, you do not have to do this translation yourself. OMNIVIEW will do it for you.

There are basically two ways to write to the screen. The first is via the E: or S: screen editor. When you activate the 80 column mode, OMNIVIEW initializes the 80 column screen and installs the 80 column E: and S: devices in the handler address table at $31A in place of the 40 column devices. Afterwards, all CIO calls to E: and S: get vectored into OMNIVIEW. This includes OPEN, CLOSE, PUT BYTE and GET BYTE. (Yes, OMNIVIEW will even go read the pixel data and figure out what character it represents!) Since every effort was made to preserve the meanings of the E: variables (ROWCRS, COLCRS, LMARGN, RMARGN, OLDCHR, etc.) even programs which manipulate them have a good chance of working in 80 columns. One difference, however, is that the logical line is only 80 characters long in 80 columns. All of this makes it easy to interface to the 80 column screen. However, there is a penalty. It is relatively slow.

This leads us to the second method of writing to the 80 column screen. There are some special hooks directly into the screen output routines of OMNIVIEW that allow a much faster screen update than is possible by going through CIO. The most important one is at $CFBA (OUTCHJ). To use this subroutine you must first calculate the exact address within the screen data of the top row of pixels of the character cell you wish to write to. Do this with the following formula: ROWCRS*240 + COLCRS/2. Put this result in MLTTMP ($66), COLCRS in the Y reg, the character to output in the A reg, and do a JSR OUTCHJ. This is exactly what OmniWriter does to update the screen in the blink of an eye. If you would like to learn more about 80 column programming, the OmniWriter source files are available for $19.95 from CDY Consulting.

Here is a memory map of the screen data area:

RAMTOP*256 ->RAMTOP HOLDS THE NUMBER OF PAGES OF RAM
RAMTOP*256-$126 ->FUTURE BUFFER FOR LAST LINE DELETED (LINBUF)
RAMTOP*256-$1F0 ->BEGINNING OF DISPLAY LIST
RAMTOP*256-$1FF0 ->BEGINNING OF DISPLAY DATA (SAVMSC)
RAMTOP*256-$2001 ->LAST BYTE OF FREE RAM (MEMTOP)

Here are the definitions of OMNIVIEW XL/XE variables:

DSTAT	$4C	USED TO SAVE STATUS
TEMP	$50	TEMPORARY REGISTER
HOLD1	$51	TEMPORARY REGISTER
LMARGN	$52	LEFT MARGIN (0-79)
RMARGN	$53	RIGHT MARGIN (0-79)
ROWCRS	$54	ROW CURSOR IS ON (0-23)
COLCRS	$55	COLUMN CURSOR IS ON (0-79); DISCERNS BETWEEN ODD AND EVEN CHARS DURING SCREEN OUTPUT (OUTCHJ)
LFTMSK	$56	INVERSE VIDEO MASK FOR EVEN COLUMNS
RGTMSK	$57	INVERSE VIDEO MASK FOR ODD COLUMNS
SAVMSC	$58	2 BYTE POINTER TO BEGINNING OF DISPLAY DATA
OLDCHR	$5D	INTERNAL FORMAT OF CHARACTER UNDER CURSOR
OLDADR	$5E	2 BYTE POINTER TO CURRENT CURSOR POSITION (ALSO SEE COLCRS) WITHIN SCREEN DATA
ADRESS	$64	2 BYTE POINTER TO CURRENT CHARACTER
MLTTMP	$66	2 BYTE POINTER WHERE NEXT CHAR WILL BE OUTPUT (ALSO SEE COLCRS) WITHIN SCREEN DATA
RAMTOP	$6A	NUMBER OF 256 BYTE PAGES OF RAM AVAILABLE
BUFCNT	$6B	BUFFER COUNT DURING E: GET CHAR
BUFSTR	$6C	RETAINS START OF LOGICAL LINE DURING E: GET CHAR (ROW/COL)
DILIST	$70	TEMP 2 BYTE PTR USED DURING GENERATION OF DISPLAY LIST
TEMP1	$79	TEMPORARY REGISTER
INSDAT	$7D	TEMPORARY REGISTER
LINBUF	$7E	2 BYTE POINTER TO A LINE BUFFER JUST PAST DISPLAY LIST
GPRIOR	$26F	PRIORITY SELECTION REGISTER
HOLD 3	$29D	TEMPORARY REGISTER
ESCFLG	$2A2	ESCAPE FLAG; USED TO DISPLAY CTRL CODES
TMPROW	$2B8	TEMPORARY STORAGE FOR ROWCRS
SCRFLG	$2BB	SCROLL FLAG; SET IF SCROLL OCCURRED
SHFLOK	$2BE	FLAG FOR SHIFT AND CONTROL KEYS
BOTSCR	$2BF	THE NUMBER OF TEXT ROWS AVAILABLE FOR PRINTING
MEMTOP	$2E5	2 BYTE POINTER TO THE TOP OF FREE MEMORY
CRSINH	$2F0	CURSOR INHIBIT FLAG; NON-ZERO TURNS CURSOR OFF
ATACHR	$2FB	LAST ATASCII CHARACTER READ OR WRITTEN
CH	$2FC	INTERNAL HARDWARE VALUE OF THE LAST KEY PRESSED
DSPFLG	$2FE	DISPLAY FLAG; NON-ZERO WILL DISPLAY CTRL CHARS
SSFLAG	$2FF	START/STOP FLAG; NON-ZERO WILL SUSPEND SCREEN OUTPUT

Use of OMNIVIEW XL/XE WITH LJK'S Letter Perfect

Any version of Letter Perfect which supports the Bit-3 board can, with the appropriate patches, be made to work with OMNIVIEW XE/XL. Some special fixed entry points were added to OMNIVIEW XE/XL to provide the necessary hooks and these can be used in your own software if needed:

CURSNJ $CFB1 TURN ON CURSOR @ OLDADR ($5E)
CURSFJ $CFB4 TURN OFF CURSOR @ OLDADR ($5E)
DELRTJ $CFB7 CLEAR TO EOL BASED UPON MLTIMP ($66) AND COL # IN REG Y
OUTCHJ $CFBA OUTPUT CHAR IN ACC TO SCREEN @ MLTIMP ($66) AND COLCRS ($55)
SCROLJ $CFBD SCROLL SCREEN UP
SCRLDJ $CFC0 SCROLL SCREEN DOWN

Here are the patches to the 80 column side of Letter Perfect Version 3.0. Use OMNIMON or any sector editor to modify a backup of the original disk (use and sector copier to make the backup). DO NOT MODIFY THE ORIGINAL DISK! For $10.00, CDY will do the patches for you. Simply send a backup copy of the 80 column side of the disk along with a check to CDY. For patches to other versions, contact CDY Consulting (214-235-2146).

SECTOR $2D BYTE $30:

WAS $EC
NOW $D9

SECTOR $2D BYTE $53:

WAS $65 EA 48 0A 0A 0A 0A 85
$64 68 4A 4A 4A 4A 85 65
NOW $EA EA 85 64 A9 00 85 65
$8A 48 20 86 25 68 AA 60

SECTOR $2D BYTE $72:

WAS $A5 EA 69 04 29 7F 85 EA
$20 86 25 20 BF 25
NOW $8A 48 20 BD CF 68 AA A5
$58 85 64 A5 59 85

SECTOR $2E BYTE $00:

WAS $20 4B 25 4C BF 25 A5 EA
$20 55 25 A9 0D 8D 80 D5
$A5 64 8D 81 D5 A9 0C 8D
$80 D5 A5 65 8D
NOW $65 4C F7 25 BF 25 A2 06
$06 64 26 65 CA D0 F9 A5
$58 18 65 64 85 64 A5 59
$65 65 85 65 60

SECTOR $2E BYTE $47:

WAS $A9 00 20 FF 25 C8 C0 50
$90 F3 60 A4 55 98 18 65
$64 85 66 A5 65 69 00 85
$67 A9 13 8D 80 D5 A5 66
$8D 81 D5 A9 12 8D 80 D5
$A5 67 8D 81 D5 A9 1F 8D
$80 D5 AD 80 D5 10 FB 60
NOW $8A 48 20 B7 CF 68 AA A0
$50 38 60 A4 55 98 4A 18
$65 64 85 66 A5 65 69 00
$D0 13 85 EA 98 48 8A 48
$A5 EA 20 BA CF EA EA EA
$68 AA 68 A8 60 85 67 60
$20 BF 25 4C 4B 25 EA 60

SECTOR $2F BYTE $00:

WAS $48 78 68 8D 85 D5 20 F9
$25 28 60 20 D2 25 A9 0F
$8D 80 D5 A5 66 8D 81 D5
$A9 0E 8D 80 D5 A5 67 8D
$81 D5 A9 09 48 A9 0A 8D
NOW $48 78 68 20 E1 25 28 60
$25 28 60 20 D2 25 20 15
$26 20 B1 CF 60 A5 66 85
$5E A5 67 85 5F 60 67 8D
$81 D5 A9 09 20 B4 CF 60

SECTOR $2F BYTE $40:

WAS $A2 10 8E 08 D5 CA 8E 80
$D5 BD CF 13 8D 81 D5 CA
$10 F4
NOW $20 01 C0 A5 58 85 9E A6
$59 CA CA 86 9F EA EA EA
$A2 FF

SECTOR $2F BYTE $72:

WAS $20 86 25
NOW $EA EA EA

OMNIVIEW XL/XE with LJK's Letter Perfect Version 3.2, 3.3

Here are the patches to the 80 column side of Letter Perfect Version 3.2, 3.3. Use OMNIMONXL of any sector editor to modify a backup copy of the original disk (use any sector copier to make the backup). DO NOT MODIFY THE ORIGINAL DISK! For $10.00, CDY will do the patches for you. Simply send a backup copy of the 80 column side of the disk along with a check to CDY. For patches to other versions, contact CDY Consulting (214-235-2146)

SECTOR $2D BYTE $39:

```
WAS SEC                                  NOW $D9
```

SECTOR $2D BYTE $5C:

```
WAS $65 EA 48 0A 0A 0A 0A 85     NOW $EA EA 85 64 A9 00 85 65
    $64 68 4A 4A 4A 4A 85 65         $8A 48 20 8F 25 68 AA 60
```

SECTOR $2D BYTE $7B:

```
WAS $A5 EA 69 04 29              NOW $8A 48 20 BD CF
```

SECTOR $2E BYTE $00:

```
WAS $7F 85 EA                    NOW $68 AA A5
    $20 8F 25 20 C8 25               $58 85 64 A5 59 85
    $20 54 25 4C C8 25 A5 EA         $65 4C 00 26 BF 25 A2 06
    $20 5E 25 A9 0D 8D 80 D5         $06 64 26 65 CA D0 F9 A5
    $A5 64 8D 81 D5 A9 0C 8D         $58 18 65 64 85 64 A5 59
    $80 D5 A5 65 8D                  $65 65 85 65 60
```

SECTOR $2E BYTE $50:

```
WAS $A9 00 20 08 26 C8 C0 50     NOW $8A 48 20 B7 CF 68 AA A0
    $90 F3 60 A4 55 98 18 65         $50 38 60 A4 55 98 4A 18
    $64 85 66 A5 65 69 00 85         $65 64 85 66 A5 65 69 00
    $67 A9 13 8D 80 D5 A5 66         $D0 13 85 EA 98 48 8A 48
    $8D 81 D5 A9 12 8D 80 D5         $A5 EA 20 BA CF EA EA EA
    $A5 67 8D 81 D5 A9 1F 8D         $68 AA 68 A8 60 85 67 60
```

SECTOR $2F BYTE $00:

```
WAS $80 D5 AD 80 D5 10 FB 60 08 NOW $20 C8 25 4C 54 25 EA 60 08
    $48 78 68 8D 85 D5 20 02         $48 78 68 20 EA 25 28 60
    $26 28 60 20 DB 25 A9 0F         $25 28 60 20 DB 25 20 1E
    $8D 80 D5 A5 66 8D 81 D5         $26 20 B1 CF 60 A5 66 85
    $A9 0E 8D 80 D5 A5 67 8D         $5E A5 67 85 5F 60 67 8D
    $81 D5 A9 09 48 A9 0A 8D         $81 D5 A9 09 20 B4 CF 60
```

SECTOR $2F BYTE $49:

```
WAS $A2 10 8E 08 D5 CA 8E 80     NOW $20 01 C0 A5 58 85 9E A6
    $D5 BD CF 13 8D 81 D5 CA         $59 CA CA 86 9F EA EA EA
    $10 F4                           $A2 FF
```

SECTOR $2F BYTE $7B:

```
WAS $20 8F 25                    NOW $EA EA EA
```


OMNIVIEW XL/XE with LJK's Letter Perfect Version 6.0 thru 6.5

Here are the patches to Letter Perfect Version 6.+. Use OMNIMONXL or a sector editor to modify a backup copy of the original disk (use any sector copier to make the backup). DO NOT MODIFY THE ORIGINAL DISK! For $10.00 CDY will do the patches for you. Simply send a backup copy of the disk along with a check to CDY. For patches to other versions, contact CDY Consulting.

SECTOR $62 BYTE $1A:

```
WAS $31 02 CA                    NOW $E6 02 EA
```

SECTOR $63 BYTE $0D:

```
WAS $65 CF 48 0A 0A 0A 0A 85     NOW $EA EA 85 64 A9 00 85 65
    $64 68 4A 4A 4A 4A 85 65         $8A 48 20 C6 08 68 AA 60
          BYTE $2C:
WAS $A5 CF 69 04 29 7F 85 CF     NOW $8A 48 20 BD CF 68 AA A5
    $20 C6 08 20 F2 07 20 85         $58 85 64 A5 59 85 65 4C
    $07 4C                           $E0 07
          BYTE $42:
WAS $A9 13 8D 80 D5 98 18 65     NOW $98 4A 18 65 64 85 66 A5
    $64 85 66 8D 81 D5 A9 12         $65 69 00 85 67 60 85 CF
    $8D 80 D5 A5 65 69 00 85         $98 48 8A 48 A5 CF 20 BA
    $67 8D 81 D5 A9 1F 8D 80         $CF 68 AA 68 A8 60 20 F2
    $D5 AD 80 D5 10 FB               $07 4C 85 07 EA EA
          BYTE $78: WAS $00      NOW $20
```

SECTOR $64 BYTE $66:

```
WAS $8D 85 D5                    NOW $20 D0 07
          BYTE $6A: WAS $4C      NOW $60
          BYTE $70:
WAS $A0 0F 8C 80 D5 A5 66 8D     NOW $20 77 08 20 B1 CF 60 A5
    $81 D5 88 8C 80 D5 A5 67         $66 85 5E A5 67 85 5F 60
```

SECTOR $65 BYTE $0C:

```
WAS $A9 A0 A0 0A                 NOW $20 B4 CF 60
          BYTE $17:
WAS $A2 10 8E 08 D5 A2 0D 8E     NOW $20 01 C0 A9 01 85 0C A9
    $80 D5 BD 02 08 8D 81 D5         $C0 85 0D EA EA EA EA EA
    $CA 10 F4 E8                     $EA EA A2 00
          BYTE $3D:
WAS $20 C6 08                    NOW $EA EA EA
          BYTE $46:
WAS $20 E3 07 29 20 F0 F9 A0     NOW $A2 06 06 64 26 65 CA D0
    $0C 8C 80 D5 A5 CF 20 8F         $F9 A5 58 18 65 64 85 64
    $07 8D 81 D5 C8 8C 80            $A5 59 65 65 85 65 60
```

SECTOR $6A BYTE $4D:

```
WAS $0D 84 0C                    NOW $0B 84 0A
```

SECTOR $79 BYTE $48:

```
WAS $42 69 74 20 33 20 66 75     NOW $4F 4D 4E 49 56 49 45 57
    $6C 6C 2D 76 69 65 77 20         $20 38 30 20 43 6F 6C 75
    $38 30                           $6D 6E
          BYTE $6B:
WAS $63 6F 6C 75 6D 6E           NOW $52 2E 49 2E 50 2E
```

OMNIVIEW XL/XE with Data Perfect Version 2.0 thru 2.5

Here are the patches to 80 column Data Perfect. Use OMNIMONXL or any sector editor to modify a backup copy of the original disk (use any sector copier to make the backup). DO NOT MODIFY THE ORIGINAL DISK! For $10.00 CDY will do the patches for you. Simply send a backup copy of the disk along with a check to CDY. For patches to other versions, contact CDY Consulting.

SECTOR $02 BYTE $06:

WAS $31	NOW $E6

SECTOR $04 BYTE $02:

WAS $30	NOW $E5
BYTE $07:	
WAS $31	NOW $E6

SECTOR $05 BYTE $1E:

WAS $42 69 74 20 33 00 33 3E	NOW $4F 4D 4E 49 56 49 45 57
$20 20 41 75 73 74 69 6E	$20 38 30 00 33 3E 20 20
$20 46 72 61 6E 6B 6C 69	$41 75 73 74 69 6E 20 38
$6E	$30

SECTOR $09 BYTE $46:

WAS $30	NOW $E5
BYTE $4B:	
WAS $31	NOW $E6

SECTOR $8F BYTE $57:

WAS $48 4A 4A 4A 4A 85 E1 68	NOW $85 E0 A9 00 85 E1 98 48
$0A 0A 0A 0A 85 E0	$20 6D 0D 68 A8 60
BYTE $6F:	
WAS $8D 85 D5 20 67 0D	NOW $20 54 0D EA EA EA

SECTOR $90 BYTE $51:

WAS $00	NOW $20
BYTE $59:	
WAS $F8 60 20 44 0D AD 83 D5	NOW $F6 60 20 44 0D BD 80 04
BYTE $65:	
WAS $A0 0F 8C 80 D5 A5 66 8D	NOW $A5 66 85 5E A5 67 85 5F
$81 D5 88 8C	$20 B1 CF 60
BYTE $7B:	
WAS $A9 20 A0 0A	NOW $20 B4 CF 60

SECTOR $91 BYTE $0D:

WAS $7B	NOW $83
BYTE $46:	
WAS $A9 13 8D 80 D5 98 18 65	NOW $98 4A 18 65 E0 85 66 A5
$E0 85 66 8D 81 D5 A9 12	$E1 69 00 85 67 60 85 56
$8D 80 E5 A5 E1 69 00 85	$98 48 8A 48 A5 56 20 BA
$67 8D 81 D5 A9 1F	$CF 68 AA 68 A8 60
BYTE $6D:	
WAS $70 50 5B 39 19 04 18 18	NOW $A0 06 06 E0 26 E1 88 D0
$78 09 20 09 00 00 A2 10	$F9 A5 58 65 E0 85 E0 A5
$8E 08 D5	$59 65 E1

SECTOR $92 BYTE $00:

WAS $CA 8E 80 D5 BD 6D 0D 8D	NOW $85 E1 60 20 01 C0 EA EA
$81 D5 CA 10 F4	$EA EA EA EA EA

USE of OMNIVIEW XL/XE with ATR8000

OMNIVIEW XL/XE has a built in terminal emulator for use with the ATR8000 which provides a serial interface for communication with the ATR and most of the standard cursor controls necessary for operation with CPM programs. The terminal emulator which will be referred to as 'ATRMON' from this point on, can be called up at any time and it is even possible to switch back and forth between the ATARI and CPM environments.

Turning on ATRMON

First of all you must activate the 80 column OMNIVIEW XL/XE screen editor (e.g., with CONTROL-A RESET). Then hold down the START, SELECT, and OPTION buttons and type any letter on the keyboard. You should hear the drive(s) reset and the ATRMON header should appear after a couple of seconds. Now put in your CPM system disk and type 'B(return)' to boot up CPM. While ATRMON is active, the START button will allow you to switch screen colors. (By the way, even in ATARI mode you can switch screen colors by holding down the START button and typing any key. This also holds true of powerup, if you press the START button after the disk boot process has begun and hold it down until the boot is finished. This allows you to change the screen colors of Letter Perfect.)

Leaving ATRMON

Leave ATRMON in almost the same way you entered it, i.e., by holding down the START, SELECT, and OPTION buttons, but this time it is not necessary to type another key. You will then see the command 'GOATARI' appear on the screen. This is to fetch the extrinsic command 'GOATARI' which is used to reset the ATR from CPM so that the drives can once more be accessed in the ATARI environment. To create this file, use DDT as follow:

1) Under CPM, insert a disk with DDT on it and type 'DDT(return) to enter DDT.
2) Type 'A100(return) JMP 0F00(return)(return) G0'
3) Back at the command level, type 'SAVE 1 GOATARI.COM(return)'

The short file 'GOATARI.COM' will have to be on any CPM disk from which you might want to return to the ATARI environment. The alternative is to reach behind the ATR and reset it whenever you return to the ATARI environment.